(12) UK Patent Application (19) GB (11) 2 312 536 (13) A

(43) Date of A Publication 29.10.1997

(21) Application No 9715829.9

(22) Date of Filing 25.07.1997

(30) Priority Data
(31) 08775841 (32) 31.12.1996 (33) US

(62) Divided from Application No 9712260.0 under Section 15(4) of the Patents Act 1977

(51) INT CL[6]
G06F 11/00

(52) UK CL (Edition O)
G4A AEX A12C A12D

(56) Documents Cited
EDGE: Work-Group Computing Report, v8, p22(1), 24 Feb 1997. Datamation, v42, n1, p49(3), 1 Jan 1996, D Baum.

(58) Field of Search
UK CL (Edition O) G4A AEX
INT CL[6] G06F 11/00 11/22 11/26
On-line: WPI, COMPUTER

(71) Applicant(s)
Erasoft Technologies Inc

(Incorporated in Canada - Alberta)

3016 5th Avenue, N.E., Suite \304, Calgary, Alberta T2A 6K4, Canada

(72) Inventor(s)
Ward Arthur Anderson
Harold Sverre Bruun Hoover
Douglas Wilfred Bird
Michael Howatt Mabey
Nashirali Samanani

(74) Agent and/or Address for Service
D Young & Co
21 New Fetter Lane, LONDON, EC4A 1DA, United Kingdom

(54) Detecting year 2000 date problems

(57) Scanning program, source and data files to determine if they involve any date formats which could potentially cause a year 2000 error.

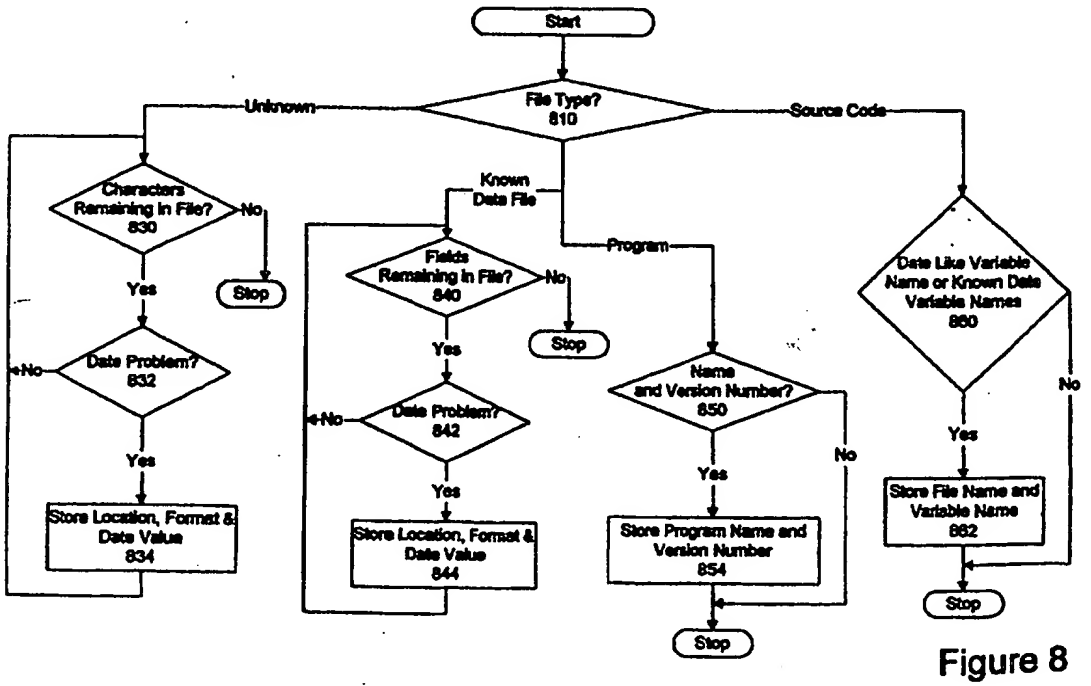

Figure 8

BEST AVAILABLE COPY GB 2 312 536 A

This print takes account of replacement documents submitted after the date of filing to enable the application to comply with the formal requirements of the Patents Rules 1995

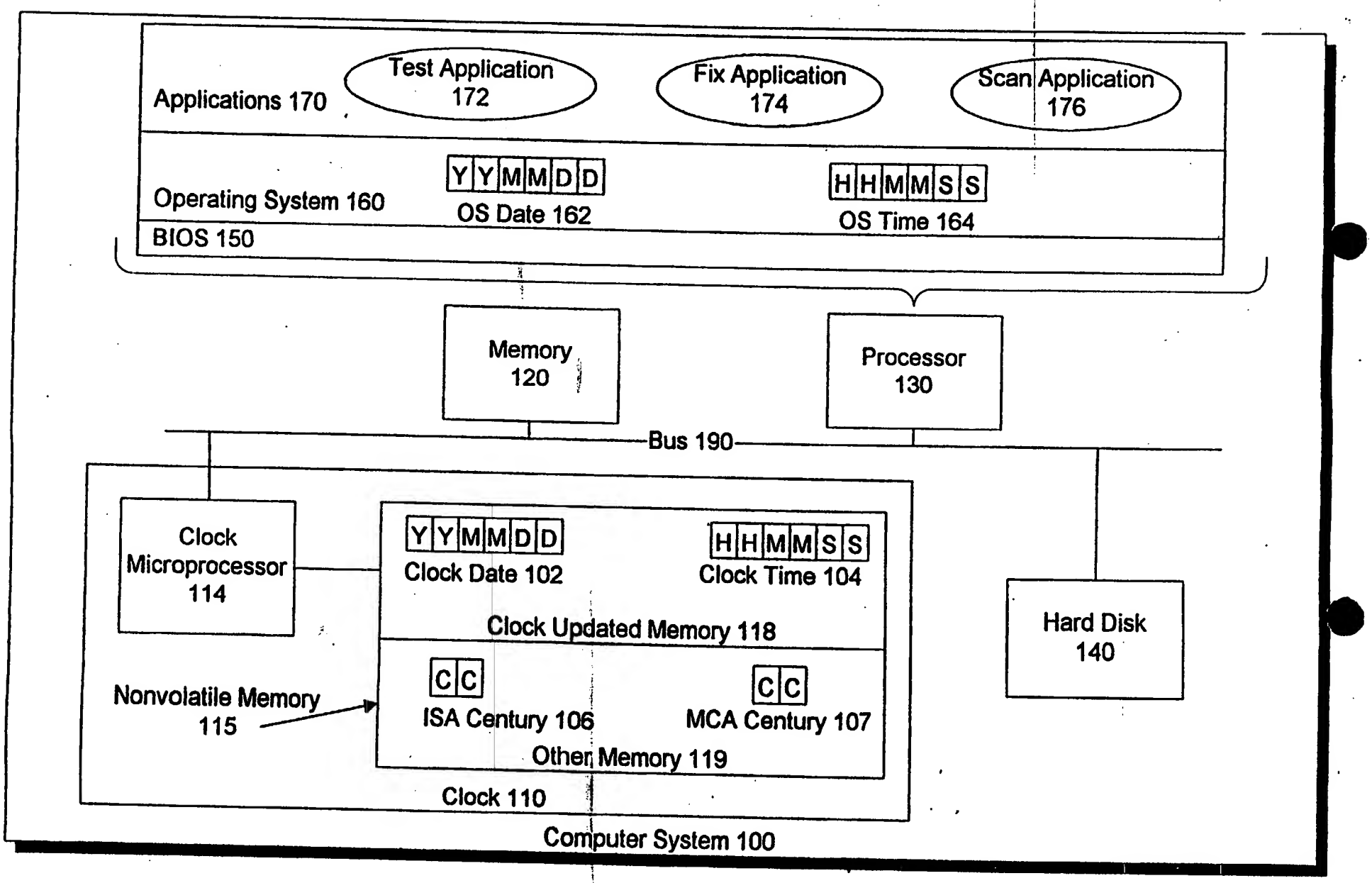
Applications 170
Test Application 172
Fix Application 174
Scan Application 176
Operating System 160
Y Y M M D D
OS Date 162
H H M M S S
OS Time 164
BIOS 150
Memory 120
Processor 130
Bus 190
Clock Microprocessor 114
Y Y M M D D
Clock Date 102
H H M M S S
Clock Time 104
Clock Updated Memory 118
Hard Disk 140
Nonvolatile Memory 115
C C
ISA Century 106
C C
MCA Century 107
Other Memory 119
Clock 110
Computer System 100

Figure 1

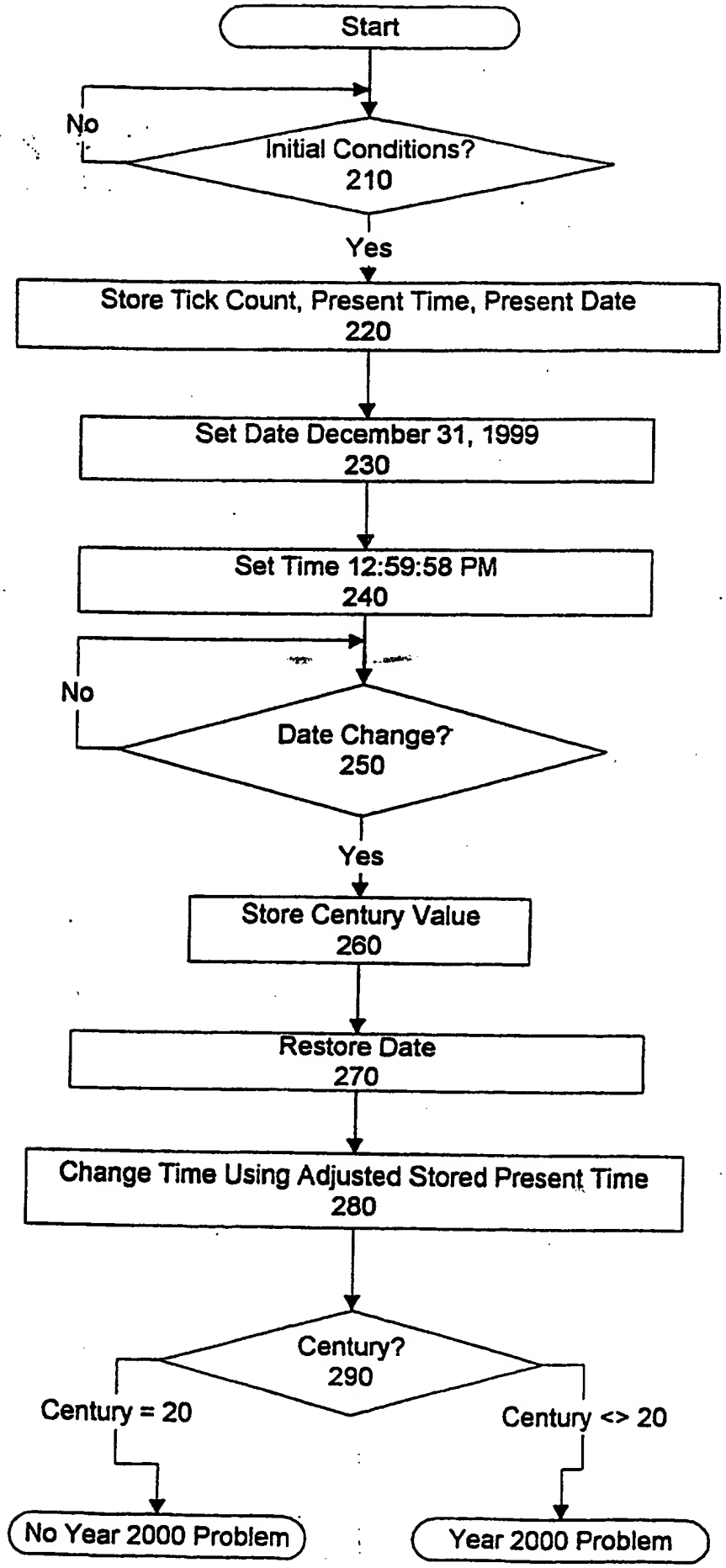
Start
No
Initial Conditions?
210
Yes
Store Tick Count, Present Time, Present Date
220
Set Date December 31, 1999
230
Set Time 12:59:58 PM
240
No
Date Change?
250
Yes
Store Century Value
260
Restore Date
270
Change Time Using Adjusted Stored Present Time
280
Century?
290
Century = 20
Century <> 20
No Year 2000 Problem
Year 2000 Problem

Figure 2

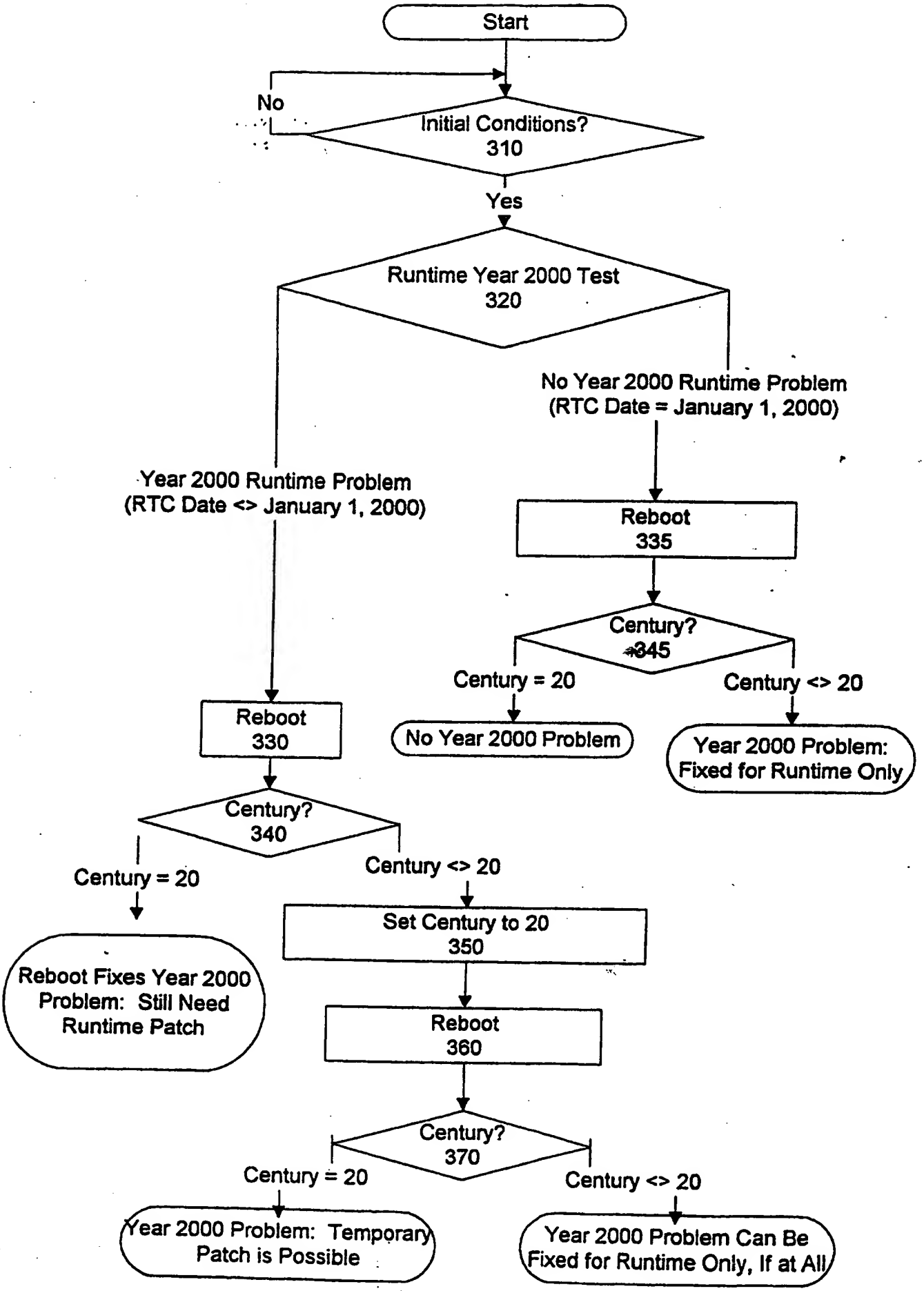
Start
No
Initial Conditions?
310
Yes
Runtime Year 2000 Test
320
No Year 2000 Runtime Problem
(RTC Date = January 1, 2000)
Year 2000 Runtime Problem
(RTC Date <> January 1, 2000)
Reboot
335
Century?
345
Century = 20
Century <> 20
No Year 2000 Problem
Year 2000 Problem:
Fixed for Runtime Only
Reboot
330
Century?
340
Century = 20
Century <> 20
Reboot Fixes Year 2000
Problem: Still Need
Runtime Patch
Set Century to 20
350
Reboot
360
Century?
370
Century = 20
Century <> 20
Year 2000 Problem: Temporary
Patch is Possible
Year 2000 Problem Can Be
Fixed for Runtime Only, If at All

Figure 3

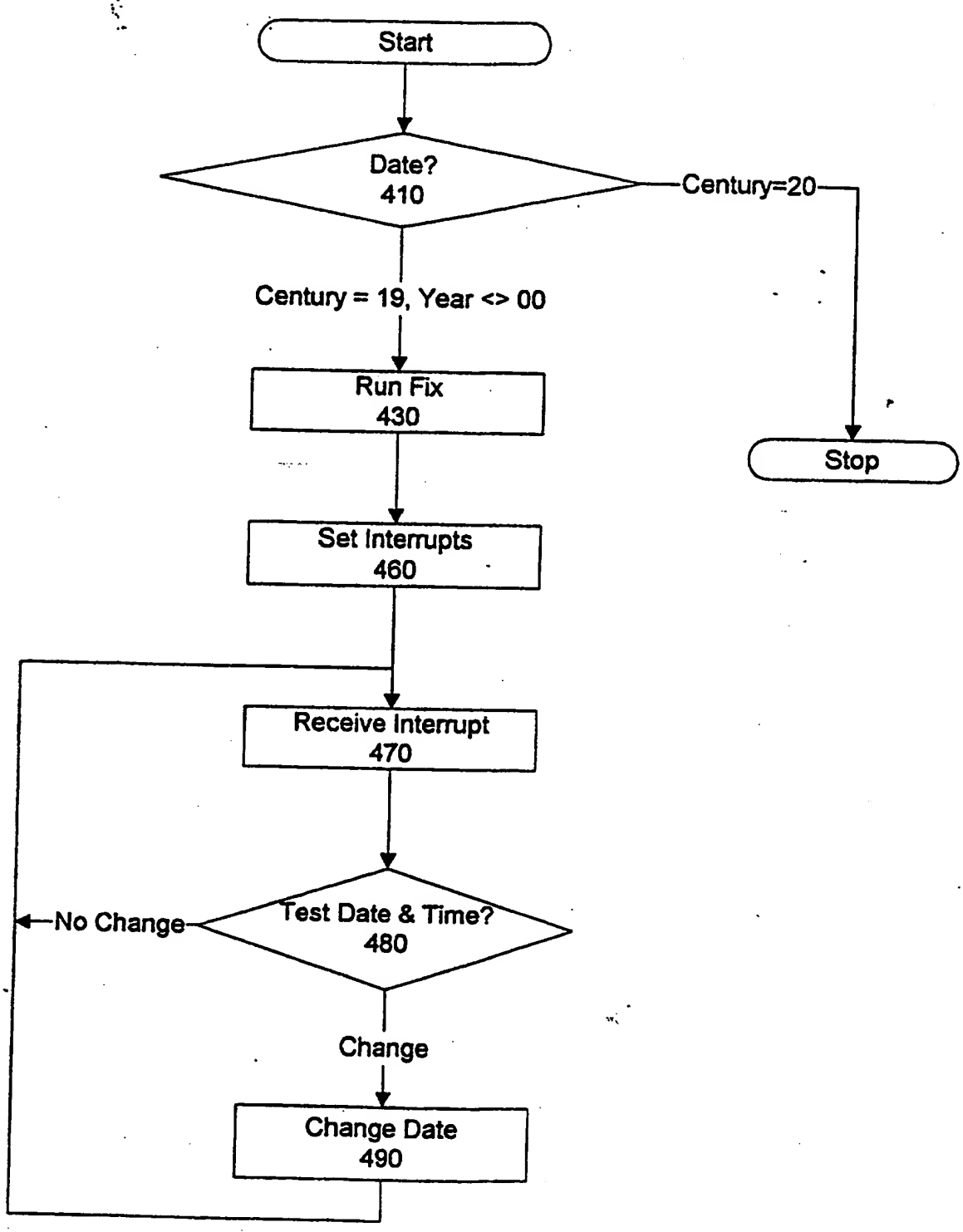
Start
Date?
410
Century=20
Century = 19, Year <> 00
Run Fix
430
Stop
Set Interrupts
460
Receive Interrupt
470
No Change
Test Date & Time?
480
Change
Change Date
490

Figure 4

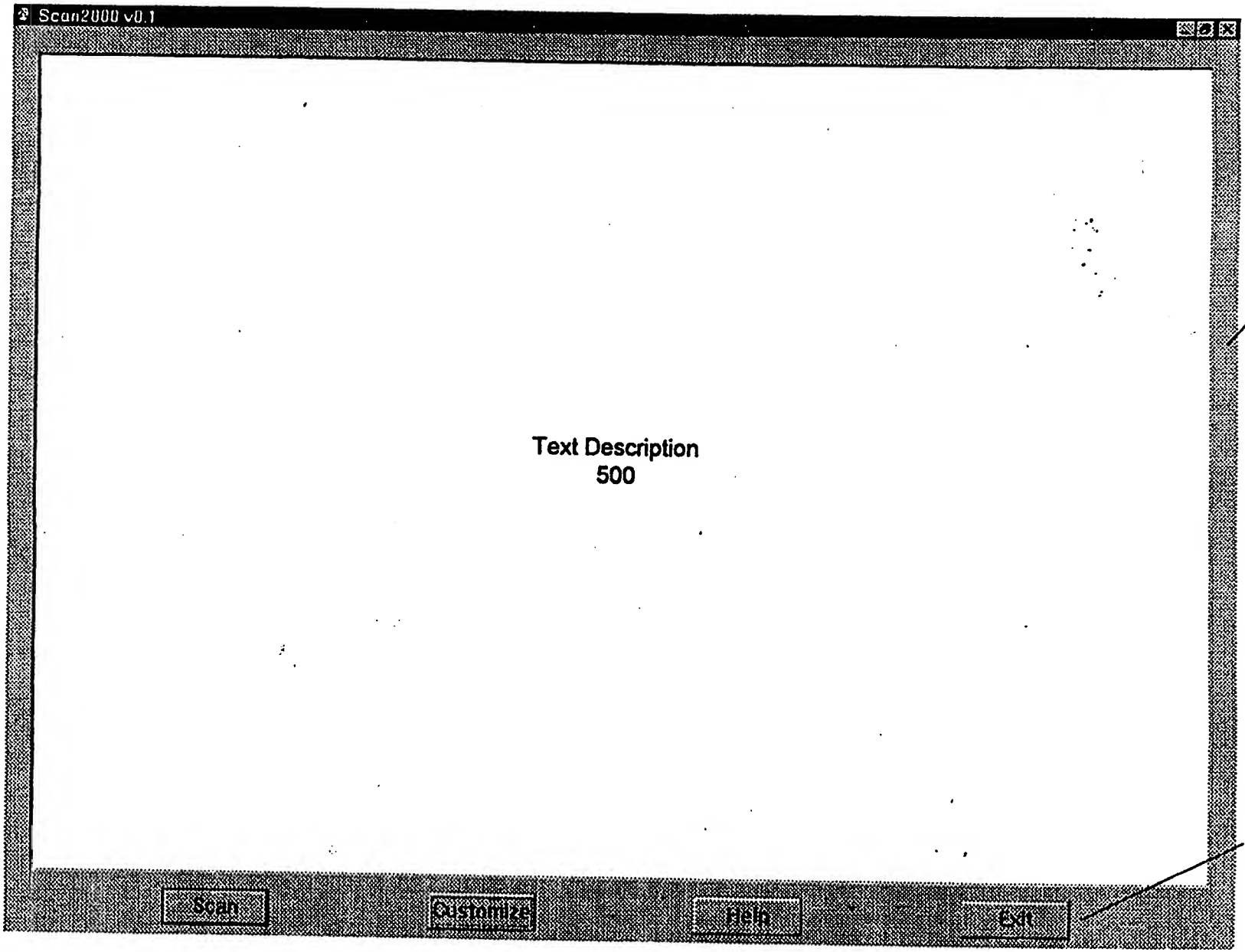
Scan2000 v0.1
Text Description
500
Scan Window
520
Menu Bar
510
Scan
Customize
Help
Exit

Figure 5

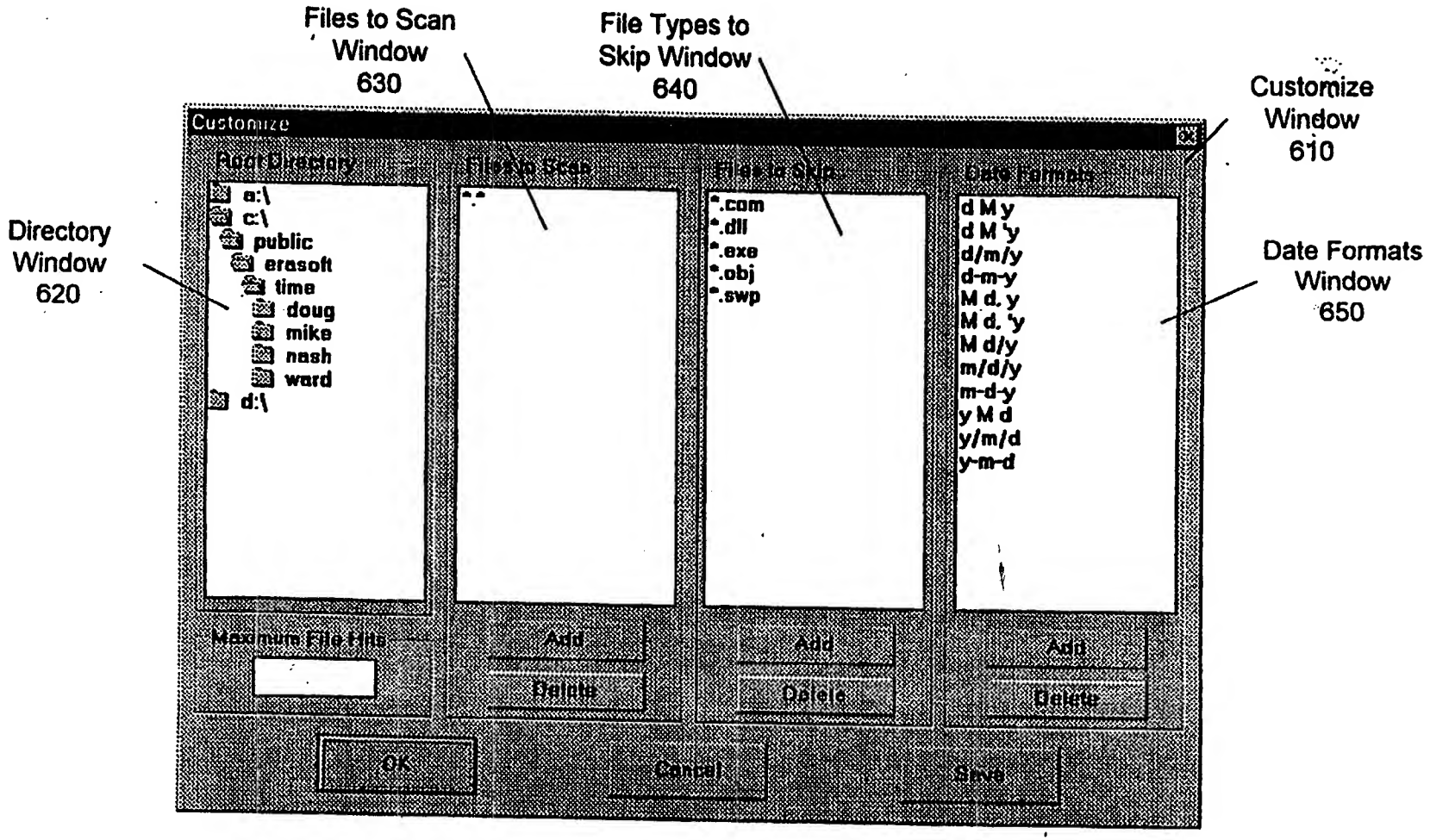
Files to Scan Window 630
File Types to Skip Window 640
Customize Window 610
Directory Window 620
Date Formats Window 650
Customize
Root Directory
Files to Scan
Files to Skip
Date Formats
a:\
c:\
public
erasoft
time
doug
mike
nash
ward
d:\
*.com
*.dll
*.exe
*.obj
*.swp
d M y
d M 'y
d/m/y
d-m-y
M d, y
M d, 'y
M d/y
m/d/y
m-d-y
y M d
y/m/d
y-m-d
Maximum File Hits
Add
Delete
Add
Delete
Add
Delete
OK
Cancel
Save

Figure 6

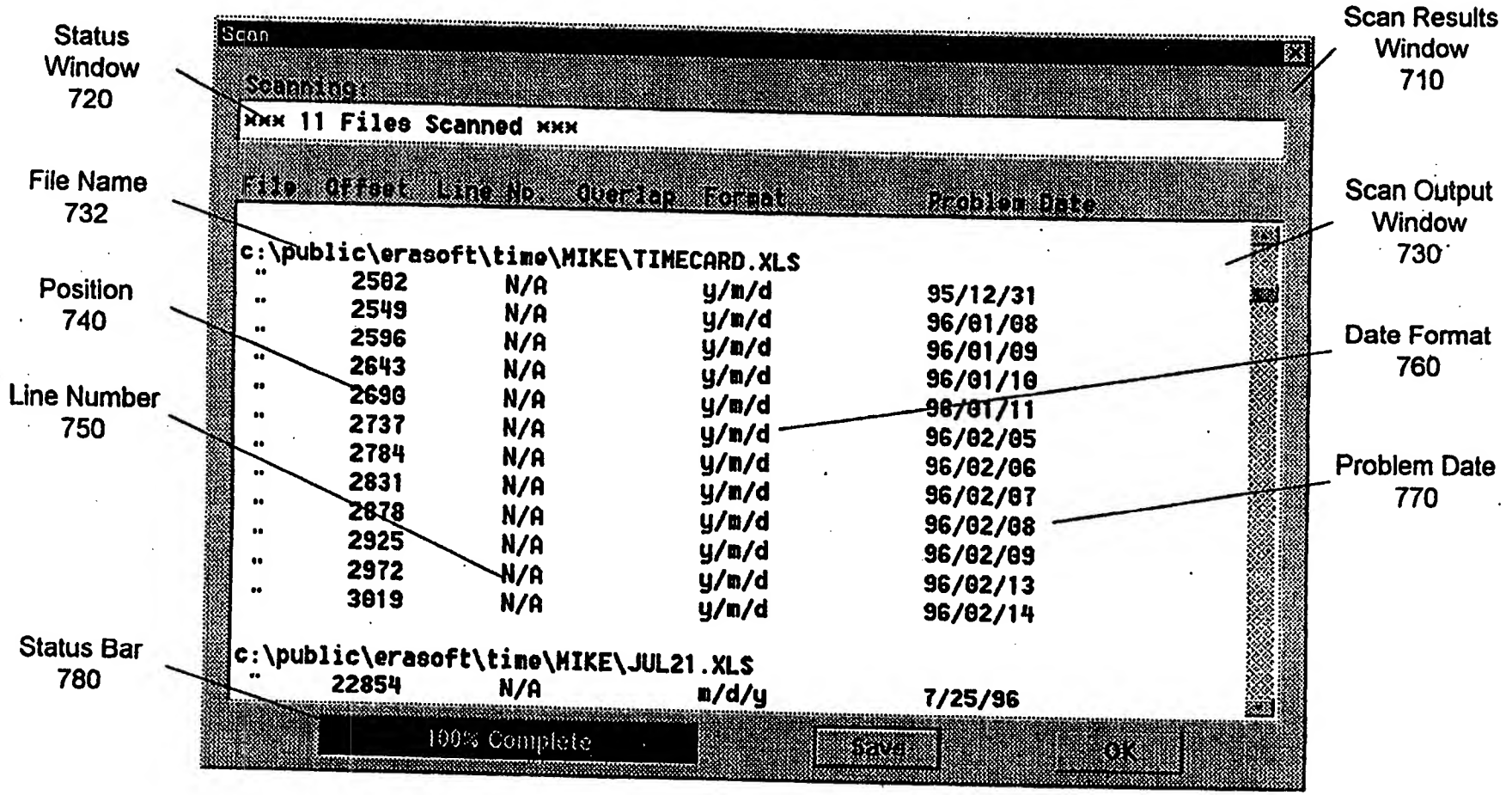
Status Window 720
File Name 732
Position 740
Line Number 750
Status Bar 780
Scan Results Window 710
Scan Output Window 730
Date Format 760
Problem Date 770
Scan
Scanning:
××× 11 Files Scanned ×××
File Offset Line No. Overlap Format Problem Date
c:\public\erasoft\time\MIKE\TIMECARD.XLS
" 2502 N/A y/m/d 95/12/31
" 2549 N/A y/m/d 96/01/08
" 2596 N/A y/m/d 96/01/09
" 2643 N/A y/m/d 96/01/10
" 2690 N/A y/m/d 96/01/11
" 2737 N/A y/m/d 96/02/05
" 2784 N/A y/m/d 96/02/06
" 2831 N/A y/m/d 96/02/07
" 2878 N/A y/m/d 96/02/08
" 2925 N/A y/m/d 96/02/09
" 2972 N/A y/m/d 96/02/13
" 3019 N/A y/m/d 96/02/14
c:\public\erasoft\time\MIKE\JUL21.XLS
" 22854 N/A m/d/y 7/25/96
100% Complete
Save
OK

Figure 7

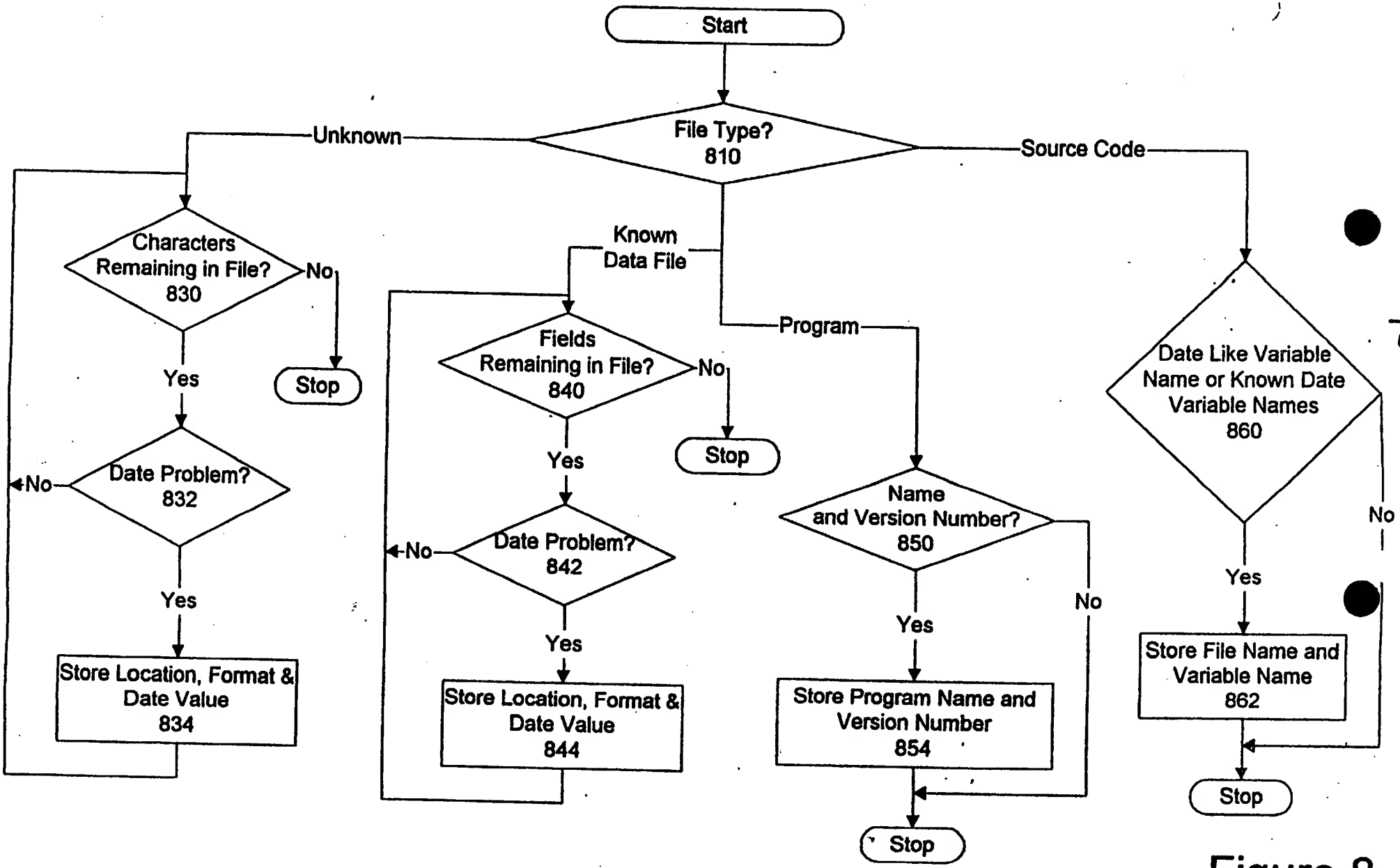
Start
File Type?
810
Unknown
Source Code
Known
Data File
Program
Characters
Remaining in File?
830
No
Stop
Yes
Date Problem?
832
No
Yes
Store Location, Format &
Date Value
834
Fields
Remaining in File?
840
No
Stop
Yes
Date Problem?
842
No
Yes
Store Location, Format &
Date Value
844
Name
and Version Number?
850
No
Yes
Store Program Name and
Version Number
854
Stop
Date Like Variable
Name or Known Date
Variable Names
860
No
Yes
Store File Name and
Variable Name
862
Stop

Figure 8

Method and Apparatus for Identifying and Correcting Date Errors

Inventors: Ward A. Anderson, Harold S. B. Hoover, Douglas W. Bird,

Nashirali Samanani, Michael H. Mabey

Background of the Invention

1. Field of the Invention

The invention relates to the field of identifying and correcting date errors. In particular, one embodiment of the invention relates to a method and apparatus for identifying year 2000 date problems in computer systems.

2. Description of Background

The transition from the year 1999 to the year 2000, as it relates to computer systems, presents a number of challenges to software development engineers and software maintenance engineers. Many operating systems and software applications require complete date and time information to function correctly. In the past, developers of some operating systems and applications chose to represent and maintain year information using only the last two digits of a complete four digit year. Where only the last two digits are used, the first two digits of the year are assumed to be '1' and '9'. This assumption can cause any number of problems in the computer system. This misinterpretation of the two digit year information is referred to as the year 2000 problem.

The following example illustrates the year 2000 problem. If a computer system is provided with the two digit year indicator "96", then the computer will assume that the year will be 1996. However, if the computer system has the year 2000 problem and the two digit year indicator is "00", then the computer system may not recognise this as the year 2000, but will substitute some other value (e.g., 1900).

The year 2000 problem affects not only large computer systems, but personal computers, such as IBM PC compatibles (PC). For PCs, the year 2000 problem can manifest itself in at least two ways: a PC's internal battery backed real-time clock (RTC) will not be updated correctly on December 31, 1999 at 11:59:59; and a PC may have any number of programs that use only the two least significant digits of the year and an implicit century value of 19.

The year 2000 problem is a technical problem, involving the operation of the computers hardware, the computer's BIOS, the computer's operating system, the format in which the date information is stored both internally within the computer's date memory and in data files, and the manner in which computer programs handle date information and communicate such information with other applications/systems. Such aspects of computer operation are not normally under an operators control, but can be critical for the operation of the computer and its application programs. The technical problem arises from the manner in which date information is represented and handled.

The following first describes the RTC related problem and then describes the PC's programs' use of two digit dates problem.

Almost all PCs include an internal battery backed real-time clock (RTC), such as the MC146818A real-time clock plus RAM device available from Motorola, Inc. Of Texas. The RTC provides the PC with a complete time-of-day clock with alarm and one hundred year calendar. The RTC keeps track of time even when the PC is off. However, the RTC does not have logic to manage and update the century information. The responsibility for managing and updating the century information is left for the system that uses the RTC.

In a PC, the operating system (OS) and/or Basic Input Output System (BIOS) are responsible for managing the century information. In many PCs today, the OS and/or BIOS manage the century information by allowing the user to access a century value stored in the RTC. The RTC includes a number of non-volatile memory locations where one byte of the non-volatile memory

store the century value. Although the BIOS and/or OS provides access to the century information, many PC BIOS's do not automatically change the century value properly for the year 2000. The BIOS of some PCs go so far as only supporting a fixed year range (e.g., from 1980 to 1999). The OS of some PCs interpret the RTC date to be January 4, 1980, if the year byte is 00.

How a program, running on a PC, requests a date value from the PC determines if and how an incorrect date value will be supplied to that program. A program can request the date value from an operating system, such as the Windows 95™ operating system, available from Microsoft, Inc., of Redmond, Washington. The Windows 95™ operating system maintains internal date values derived from the date values within the RTC. If the century date value is incremented to 20 when the year transitions from 99 to 00, the programs that use the operating system's date value receive a correct date. This holds true until a reboot occurs in the year 2000 on a system with an incorrect RTC value. However, not all programs use the date value supplied from the operating system. Programs can also retrieve the date value directly from the RTC using IN and OUT instructions or indirectly from the RTC using the BIOS (e.g., using Time-of-Day interrupt functions). These programs may receive the wrong date value in the year 2000, and subsequent years.

A test to determine whether a given PC has the RTC year 2000 problem is described in the YEAR2000.TXT text file associated with the YEAR2000.EXE program, both available from Air System Technologies, Inc. of Dallas Texas. In the described test a user is required to perform a number of steps to determine whether his/her PC has the year 2000 problem. However, the test is not automatic, making the test simply inaccessible for some users. That is, because the user is required to perform a number of steps, many users will simply not perform the steps. This is a

less than desirable result. Additionally, the information provided in the described test does not provide the user with all the information relating to a potential RTC year 2000 problem.

Once a personal computer has been identified as having the RTC year 2000 problem, a number of possible solutions have been used.

The first solution is to let the user correct the date problem directly by having the user enter a DOS command to set the date on January 1, 2000. The date will be incorrect, though, until the correct date is set.

Another solution is to let the operating system and/or BIOS attempt to fix the problem. Some operating systems and/or BIOS will fix the date problem by changing the century value in the real-time clock when the computer boots up. For example, the BIOS examines the information in the RTC and, if the RTC determines that the century should be "20" instead of "19" then the BIOS will update the century value in the RTC. However, fixing the century upon boot up is an inadequate solution to the RTC year 2000 problem for many PCs. If the computer is running during the year 2000 transition (e.g., the PC is running at 11:59:59 on December 31, 1999), then the RTC will have the incorrect century value until the computer is rebooted. Any programs that access the RTC's century value directly, prior to rebooting, will receive an incorrect century value. This type of error could cause significant problems in PCs used in control systems and the like that run constantly.

Another solution is to use the YEAR2000.EXE program available from Air Systems Technologies, Inc. of Dallas, Texas. The YEAR2000.EXE program uses standard periodic interrupts to test for a year change upon boot up and when the two digit year value in the RTC changes. However, the YEAR2000.EXE appears to only be invoked via a single interrupt chain

that can easily be corrupted by other programs. Thus, the YEAR2000.EXE may not work in some user environments.

Therefore, it is desirable to provide a user with an automated method of testing a personal computer for the RTC year 2000 problem and applying a fix that has a low likelihood of either interfering with other programs or of being interfered with by other programs.

PC applications that maintain a year date value using the two least significant digits of a complete date value are prone to failure. When the century is implied or not considered in date calculations, application results may be incorrect or the application may fail. Often a user will not know if a particular PC program uses only two digits to represent a year value. Therefore, it is desirable to inform the user of potential year 2000 problems in programs on the PC.

Summary of the Invention

A method of detecting year 2000 date problems in a computer system, and correcting some of those problems, is described. One embodiment of the invention includes a computer system executing a program for testing and correcting, if possible, incorrect century values in a real time clock included in the computer system. This embodiment also includes a program for scanning files on the computer system and noting which files may have year 2000 type problems in the files.

Another embodiment of the invention includes a method of automatically detecting a date problem in a computer system. The computer system includes a processor, a memory, a clock, and a nonvolatile memory. The processor is coupled to the memory, the clock and the nonvolatile memory. The nonvolatile memory stores a clock date value and a clock time value. The clock date value includes a clock century value and a clock year value. The computer system also includes a program that executes at least some of the following steps in the method. At a first time, store a first date value and a first time value corresponding to the clock date value and the clock time value, respectively. Change the clock date value to December 31, 1999. Change the clock time value to a value less than or equal to 11:59:59 PM. Test the clock century value. If the clock century value indicates twenty, then indicate that the date problem does not exist. If the clock century value does not indicate twenty and the clock year value does not indicate 99, then indicate that the date problem does exist. Change the clock date value to correspond to the first date value. Change the clock time value to correspond to the first time value.

Another embodiment of the invention includes another method of detecting a date problem in a computer system. The computer system includes a processor, a memory, a clock, and a

nonvolatile memory. The processor is coupled to the memory, the clock, and the nonvolatile memory. The nonvolatile memory includes a clock date value. The clock date value includes a clock century value. The method comprise the following steps. The computer system causes the clock century value to indicate twenty. The computer system restarts the computer system. The clock century value is tested after the restart of the computer system. If the clock century value does not indicate twenty, then the computer system indicates that the date problem exists.

Another embodiment of the invention includes a method of fixing a date handling problem in a computer system. The computer system includes a processor, a nonvolatile memory and a clock. The processor is coupled to the nonvolatile memory and the clock. The nonvolatile memory has a century field and a year field. The clock modifies the century field and the year field according to the time. The method comprises the following steps. Select an interrupt from a set of predetermined interrupts. Set the interrupt to cause the clock to interrupt the processor at a predetermined time. Receive the interrupt. After receiving the interrupt, test the century field and the year field. If the century field does not indicate twenty and the year field indicates a year less than a predetermined value, then change the century field to equal twenty.

Although many details have been included in the description and the figures, the invention is defined by the scope of the claims. Only limitations found in those claims apply to the invention.

Brief Description of the Drawings

The figures illustrate the invention by way of example, and not limitation. Like references indicate similar elements.

Figure 1 illustrates a computer system including one embodiment of the invention.

Figure 2 illustrates one embodiment of a method of testing a computer for the year 2000 problem.

Figure 3 illustrates another embodiment of a method of testing a computer for the year 2000 problem.

Figure 4 illustrates one embodiment of a method of correcting the year 2000 problem.

Figure 5 illustrates one embodiment of a user interface to a program for scanning files for the year 2000 problem.

Figure 6 illustrates one embodiment of a scan test for the year 2000 problem.

Figure 7 illustrates some example results of a scan test for the year 2000 problem.

Figure 8 illustrates one embodiment of a method of scanning for the year 2000 problem.

Description of Embodiments of the Invention

Overview

One embodiment of the invention includes a personal computer executing a suite of programs. In this embodiment, the suite of programs are used together to detect and correct year 2000 problems in a personal computer. In another embodiment of the invention, only certain parts of the suite are included.

The suite of programs first test whether a year 2000 problem exists in the personal computer's real time clock (RTC). The RTC tests include both a run-time test and a reboot test. The run-time test sets the RTC to December 31, 1999 and to a time close to 11:59:59 PM. The run-time test begins testing the date value to determine whether the personal computer rolled the date over correctly. If the date rolled over successfully (i.e., January 1, 2000), then the personal computer may not have the year 2000 problem. If the date does not roll over properly (e.g., not January 1, 2000), then the personal computer has the year 2000 problem. The reboot test determines whether the personal computer will fix the century problem upon reboot and/or whether any fix applied to the century value will remain after a reboot. If a year 2000 problem is detected, and the year 2000 problem can be fixed, then the suite of programs installs a fix program in the personal computer. The fix program monitors the date and time of the personal computer and corrects the century value when the personal computer rolls over to the year 2000. The suite of programs also tests for potential year 2000 problem data files and programs. That is, the suites of programs scan data files and programs on the personal computer for potential year 2000

problems. The scan involves scanning all of the data files in the personal computer for known problem date formats, scanning known data file types for fields that are problem fields, and searching for programs with known year 2000 problems. In other embodiments, not all of the scan tests are performed.

In another embodiment of the invention, some of the programs in the suite are run from a network computer on the personal computers on the network. The suite of programs includes a database of known problem computers and as part of the testing, the suite of programs detects the known problem computers and feeds this information back to the user.

The following first describes the computer system including one embodiment of the invention, then describes each of the programs in the suite, and then describes additional embodiments of the invention.

Computer System

Figure 1 illustrates a computer system 100 including one embodiment of the invention. This section first lists the elements of Figure 1 then their operation.

The computer system 100 includes a processor 130, a memory 120, a clock 110, a hard disk 140, all being coupled to a bus 190. The clock 110 includes a clock microprocessor 114 coupled to a nonvolatile memory 115. The nonvolatile memory 115 includes a clock updated memory 118 and an other memory 119. The clock updated memory 118 includes storage locations for a clock date value 102 and a clock time value 104 and clock alarm values 103. The clock date value 102 includes a two digit year value, a two digit month value, and a two digit day value. The clock time value 104 includes a two digit hour value, a two digit minute value, and a two digit second

value. The clock alarm values 103 include an alarm seconds value, an alarm minutes value, and an alarm hours value. The remaining memory locations in the other memory 119 are available to the BIOS to use for other purposes such as storage locations for an Industry Standard Architecture (ISA) century value 106 and a Micro Channel Architecture (MCA) century value 107. The computer system 100 includes a number of software components. The software components include a Basic Input Output System (BIOS) 150, an operating system 160, and a number of programs 170 (also referred to as applications 170). The operating system includes an OS date value 162 and an OS time value 164. The applications 170 include a test application 172, a fix application 174 and a scan application 176.

The following now describes each of the above elements in greater detail and how the elements interact.

In one embodiment of the invention, the computer system 100 includes an IBM compatible personal computer, such as a HP Vectra 166 Pentium computer available from Hewlett-Packard, Inc. of Mountain View, CA. In one embodiment, the computer system 100 includes a display device, such as a cathode ray tube or a flat panel display. In one embodiment of the invention, the suite of applications test the computer system 100 for year 2000 problems and install fixes where appropriate.

The processor 130 is an x86 compatible microprocessor, such as a Pentium™ microprocessor available from Intel, Inc. of Santa Clara, CA. The processor 130 executes instructions in the memory 120 to interface with the other elements in the computer system 100.

The memory 120 includes read only memory and/or random access memory. The memory 120 stores data and instructions for the processor 130.

The clock 110 includes a battery backed real time clock (RTC), such as, the MC146818A real-time clock plus RAM device available from Motorola, Inc. of Texas. The clock microprocessor 114 maintains the information in the clock updated memory 118, even when the computer system 100 is powered off. In particular, the clock microprocessor 114 constantly updates the clock data value 102 and the clock time value 104. The clock microprocessor 114 also provides access to the clock updated memory 118 and the other memory 119 to the processor 130 or other devices coupled to the bus 190. Thus, the processor 130 can read and modify the clock date value 102 and the clock time value 104. Reading and modifying the clock date value 102 and the clock time value 104 is an important feature for the test application 172 and the fix application 174. The test application 172 uses the read and modify feature to test whether the computer system 100 properly deals with the year 2000. The fix application 174 monitors the clock date value 102 using the read feature and corrects the year 2000 problem using the modify feature.

Note that the BIOS 150 may use one of two locations for the date century byte, depending on the bus 190 architecture. This illustrates that for any given computer system 100, the century value will be in one location where the bus 190 is an ISA bus and in another location where the bus 190 is a Micro Channel Architecture bus. However, for any given computer system 100, only one century value is maintained in the nonvolatile memory 115.

The hard disk 140 includes a hard disk drive, such as is available from Seagate, Inc. of Scotts Valley, CA. The hard disk 140 stores data and program files that are scanned by the scan application 176. The hard disk 140 represents both internal and external hard disk drives or other

permanent storage devices such as tape drives, optical drives, removable storage devices, network storage devices, and flash memory.

The BIOS 150 is a software program that provides the operating system 160, and any applications 170, with an interface to the hardware elements in the computer system 100. In particular, the BIOS 150 simplifies the interaction with the clock 110 by the operating system 160 and the applications 170.

The operating system 160 is a software program that provides applications with a higher level interface to the hardware elements in the computer system 100 and to the BIOS 150. One example of the operating system 160 is the DOS 6.0™ operating system available from Microsoft Corporation of Redmond Washington. Other embodiments of the operating system 160 include the Windows 95™ and the Windows NT™ operating system, and Windows 3.1™ and Windows 3.11™ windowing systems. Importantly, in one embodiment, the operating system 160 can supply the applications 170 with the OS date value 162 and the OS time value 164, which may not exhibit the year 2000 problem. However, the test application 172 is able to automatically test the clock date value 102 in the clock 110.

The test application 172 automatically tests the computer system 100 for the year 2000 problem in the clock 110. In one embodiment, the test application includes a C language program. The C language program can execute under the DOS, Windows 3.1, Windows 3.11, Windows 95, Windows NT, and OS/2 environments. Thus, one embodiment of the invention can automatically test for the presence of the year 2000 problem in the clock 110 of the computer system 100 under a number of different environments.

One embodiment of the test application 172 completely controls the computer system 100, causing potentially multiple reboots of the computer system 100. In one embodiment, this is accomplished by storing the test application 172 on a floppy disk and inserting the floppy disk into the computer system's floppy disk drive (not shown). The computer system will then reboot from the floppy disk drive, allowing the test application 172 to control the computer system 100. In one embodiment, the test application 172 automatically installs the fix application 174 if the year 2000 problem is detected in the computer system 100.

The fix application 174 fixes the year 2000 problem in the clock 110 where applicable. That is, if the computer system 100 does not have a particular type of year 2000 problem, the fix application 174 will not be installed. In one embodiment, the fix application 174 an installation program, for installation, and an assembly language terminate-and-stay-resident (TSR) program, for fixing the year 2000 problem in the clock 110. The fix application 174 can also execute under a number of environments.

The scan application 176 scans files in the hard disk 140, or elsewhere in the computer system 100, for potential year 2000 problems in software source code files, data files, and executable files. In one embodiment, the scan application 176 includes a C++ language program that executes under a number of environments.

Each of these applications 170 is described in greater detail below. Importantly, in combination, the applications 170 can detect many year 2000 problems in personal computers and fix some of those problems.

Run-time Test

Figure 2 illustrates one embodiment of a method of testing a computer system 100 for the year 2000 problem. The purpose of the embodiment of Figure 2 is to perform a run-time test to determine whether computer system 100 properly updates the century value and the clock date value 102 and clock time value 104 for the year 2000 transition. Generally, the embodiment automatically sets the clock date value 102 and the clock time value 104 near the turn of the century, then observes the year 2000 transition watching for any problems. The embodiment of Figure 2 is automatically executed by the test application 172 on the computer system 100.

As with the other embodiments of methods used in the invention, the methods described are illustrative and not meant to limit the invention. For example, although the embodiments are shown as a number of steps performed in a particular order, the order of the steps is not necessarily fixed. In other embodiments, some steps are performed in parallel, while other steps are performed in an order different than is presented.

The first step 210 tests whether an initial set of conditions is present. In one embodiment, this includes testing to determine that the computer system 100 includes the clock 110 (if none is present, this is reported to the user). Testing for the existence of the clock 110 is accomplished by calling software interrupt x15 (BIOS System Functions), function xC0 (Return System BIOS Configuration). The call provides a pointer to ten bytes of memory with various computer system 100 configuration information. The RTC present flag is contained in the sixth byte (Feature Information Byte 1), fifth bit (RTC Is Present Bit). The step 210 also determines the type of the

bus 190 (e.g. MCA or ISA). This can be used to determine where the century value is stored in the other memory 119.

Optionally, the step 210 will wait for a period of time until the least significant digit in the seconds field of the clock time value 104 is a zero. Waiting has two advantages. First, waiting allows the test to be timed in such a way as to preclude the need to calculate a date rollover when the time is reset and adjusted at test completion. Second, waiting removes the need to perform relatively complicated binary coded digit (BCD) math to return the clock time value 104 to the correct time after the test is complete. In another embodiment, the test waits only if the seconds field of the clock time value 104 is close to rollover (e.g., no waiting occurs if the seconds field is in the range 0-55). In another embodiment, the test application 172 includes routines for performing a date and time rollover and the waiting part of step 210 is not included. In another embodiment, the test application 172 will restore the system time without consideration for the test duration and again the waiting part of step 210 is not included.

After the initial conditions are set up, then step 220 is performed. Step 220 stores information to allow the test application 172 to restore the clock 110 to the correct date and time after the testing is performed. In step 220, the present values of the clock date value 102, clock time value 104 and the century value are stored. Also, a "tick count" is stored. The tick count is a running count of the number of clock ticks which is stored by the BIOS 150 and updated through the interrupt handler for the interrupt x08. A clock tick is approximately 54.9 ms. In one embodiment, the tick count is stored by accessing the Timer Tick Counter from BIOS 150 data location x406C and stored in a local valuable in the test application 172. The stored tick count is

used by the test application 172 to compensate for the time taken to perform the test when restoring the present clock time value.

At step 230, the test application 172 sets the clock date value 102 to December 31, 1999, using interrupt x1A (Time-Of-Day), function x5 (Set CMOS Date).

At step 240, the test application 172 sets the clock time value 104 to 11:59:58 PM, using interrupt x1A (Time-Of-Day), function x3 (Set CMOS Time). A two second grace period is allotted to accommodate the internal hardware logic that processes the date changes. Modifying the contents within the clock 110 within two seconds of a month rollover may produce undesirable results. In another embodiment, where the undesirable results are less likely to occur, the time is set to 11:59:59 PM. In other embodiments, other time periods are used (e.g., 3 seconds before midnight).

At step 250, the test application 172 begins sampling the date information stored in the clock 110. In one embodiment, this is performed by sampling the clock time value 104 using the interrupt x1A (Time-Of-Day), function x2 (Read CMOS Time). The sampling continues until the hours value is zero, the minutes value is zero, and the seconds value is x02. Other embodiments of the invention use different waiting periods (e.g., 0 seconds, 3 seconds).

At step 260, the century value is stored.

At step 270, the present clock date value, stored in step 220, is restored.

At step 280, the present clock time value, stored in step 220, is restored. However, in one embodiment, in step 280, the clock tick count is accessed. The difference between the tick count stored in step 220 and the present tick count is determined. The number of elapsed seconds are calculated from this difference. These seconds are then added to the present clock time value to compensate for the time taken to run the test. The new present clock time value is then stored into the clock updated memory 118.

At step 290, the century value indicates whether the computer system 100 has a run-time year 2000 problem. In particular, if the century value does not equal x20, then the personal computer has the year 2000 problem.

Thus, the test application 172 indicates to the user whether the computer system 100 properly updates the clock 110 during the transition from December 31, 1999 11:59:59 PM to January 1, 2000, 00:00:00 and while the computer system 100 is running. The testing described in relation to Figure 3 tests the computer system 100 for updating the clock 110 century value when the computer system 100 is not running during the transition.

Reboot Test

Figure 3 illustrates another embodiment of a method of testing a computer for the year 2000 problem. The purpose of the embodiment of Figure 3 is to perform a test to determine whether the computer system 100 properly updates the century value, the clock date value 102 and clock time value 104 after a reboot, and whether a change to the century value will remain after the reboot. Generally, the embodiment automatically sets the clock date value 102 and the clock time value 104 near the turn of the century, then observes the year 2000 transition watching for any

problems. The test application 172 reboots the machine, depending upon certain factors, performs additional tests, and reports the results of the test. The embodiment of Figure 3 is automatically executed by the test application 172 on the computer system 100.

At step 310, a number of initial conditions are established. In one embodiment, the test application 172 waits to perform the test if the current clock time value 102 is close to midnight (e.g., 23:59:57). The current time and date is also saved in step 320.

At step 320, the testing technique described in step 230 through step 260 is performed. If the century value is not x20, then a year 2000 problem exists in the computer. Further testing is performed to determine if the year 2000 problem can be fixed, see step 330 onward. If the century value is x20, then no year 2000 transition problem exists, but further testing may be optionally performed to ensure that upon reboot, the year 2000 problem does not arise, see step 335 onward.

At step 330, the computer system 100 has shown that it does not update the century value properly in the clock 110. Now the test application determines whether the BIOS 150, operating system 160, or other applications 170 will fix the century value upon reboot.

At step 340, the test application 172 tests the century value after the reboot step 330. If the century value is x20, then the century value was corrected during the reboot, and this is reported to the user. The user can then optionally apply the fix application 174. If the user does not want to apply the fix application 174, then the computer system 100 will not update the century value in the clock 110 until after a reboot. This may be an acceptable alternative for some users and this part of the test provides important information for the user.

At step 350, the century value tested in step 340 is not x20, and it is not known if such a century value will be allowed by the system, or if this value will be corrupted during a reboot. So at step 350 x20 is stored into the century value.

At step 360, the computer system 100 is rebooted.

At step 370, the century value is tested again. If the century value is x20, then the fix application 174 will be able to correct this value automatically. If the century value is not x20, then the fix application 174 will not be able to correct the clock value.

Returning to step 320, the path where the computer system 100 has shown that it properly handles the year 2000 problem is now discussed. For completeness, it is valuable to ensure that the clock date value 102 stays properly set even after a reboot. As noted above, however, these steps are optional and are not included in some embodiments of the invention.

At step 335, the computer system 100 is rebooted by the application.

At step 345, the test application 172 tests the century value. If the century value is x20, then no year 2000 problem occurs in the system and no fix application 174 need be installed. If the century value is not x20, then the year 2000 problem reoccurs at start up and only a run-time fix can be made, if at all.

After the testing is done, the current clock time value 102 stored in step 310 is stored back into the clock 110. In another embodiment, the current clock time value 102 stored in step 310 is modified prior to storing it back into the clock 110.

Year 2000 Run-time Fix

Figure 4 illustrates one embodiment of a method of correcting the year 2000 problem. The purpose of the embodiment of Figure 4 is to monitor the clock date value 102 and fix the century value after the year 2000 transition. The embodiment of Figure 4 automatically fixes the year 2000 transition problem in the clock 110. A user can selectively install the fix application 174. However, in another embodiment, the results of the test application 172 cause the fix application 174 to be installed automatically.

In one embodiment, the fix application 174 includes a portion executed in the AUTOEXEC.BAT and the INSTALL.BAT (called the non-resident program) and another portion which is an executable TSR program (called the resident program). Step 410 through step 430 correspond to the AUTOEXEC.BAT portion, step 460 corresponds to the INSTALL.BAT portion, and step 470 through step 490 correspond to the TSR program. Other embodiments perform the steps in different programs (e.g. all the steps are performed in the TSR program). The non-resident program performs validation checks, establishes and initializes the timing and interrupt handling mechanisms and makes the resident program memory resident. The resident program monitors the clock date value 102 and takes the appropriate action when the year 2000 transition occurs.

The non-resident program first ensures that the system configuration information is available, then tests for the presence of the clock 110, and tests to see if the fix application 174 has already been installed. If the system configuration information is not available, the clock 110 is not

present, or the fix application 174 has already been installed, then the non-resident program does not proceed.

At step 410, the clock date value 102 is checked. If the century value is x20, then the year 2000 transition has occurred and the resident program need not be executed. If the century value is x19 and the year value is less than x80, then the century value is set to x20 and the resident program is not installed. Otherwise, if the century value is x19, then the resident program continues onto step 430. Step 410 is optional. In some embodiments of the invention, the non-resident program always installs the resident program.

At step 4 30, the non-resident program runs a non-resident install program. The non-resident install program performs step 460. In one embodiment, the non-resident program executes step 430.

At step 460, the interrupts are set. In one embodiment, multiple interrupts (e.g. three interrupts) are set. In another embodiment, interrupts are selected to be set from a set of interrupt types (the selection can be performed by the computer or can be performed by user). Setting multiple interrupts has the advantage of still correcting the date when one or two interrupts are corrupted by other applications 170. That is, if at any time during the execution of the operating system another application corrupts one, or even two of the interrupts, the resident program will still correctly update the century value. Setting the interrupts is described in greater detail below. After the interrupts are set, the non-resident program returns control to the operating system 160.

Regardless of the interrupt types set, an interrupt will be received by the resident program, step 470. This activates the fix application 174's test and patch features.

At step 480, the resident program tests the clock date value 102 to determine whether the year 2000 transition has occurred since the last interrupt. If the year 2000 transition occurred (e.g., the clock date value is x1900), then, in step 490, the clock date value 102 changed to the year 2000. If the year 2000 transition has not occurred, then the clock date value is not changed. In either case, the resident program returns to step 440. In one embodiment, if the century value is not x20 but the year value is less than x80, step 490 is executed and the century value is changed to x20. After step 490, the non-resident program returns to step 470.

Prior to describing the interrupt types, an alternative embodiment of the invention is described. In this embodiment, at step 460, one or two of the three possible interrupts are selected for setting in step 470. Note that each interrupt type has particular advantages and disadvantages which may be weighed when making an interrupt type selection. Also, because this embodiment is able to select among the different interrupt types, this embodiment of the invention can more easily avoid conflicts with other programs.

Each of the interrupt types is now described. The first type of interrupt uses the User Timer Tick Counter available through software interrupt x1C. This interrupt is activated approximately eighteen times per second. Adding the resident program to the list of interrupt service routines for interrupt x1C enables the resident program to be interrupted approximately eighteen times per second. However, in one embodiment, testing the clock date value 102 at eighteen times per

second is undesirable. Therefore, for this interrupt type, at step 480, the resident program does not test the clock date value 102 for every interrupt, but only after a certain number of interrupts (e.g., every sixteenth interrupt). One problem with this interrupt is that other applications may also use this interrupt. This leads to a chain of interrupt service routines that must be called for every User Timer Tick Counter interrupt. Each interrupt service routine is responsible for calling the next service routine in the chain. This chain can become corrupted. Therefore, in the embodiment where step 460 selects which interrupts to set, if this interrupt is already set by another program, or if testing indicates that this chain may become corrupted, then another interrupt is used.

The second type of interrupt is the clock 110 Alarm available through software interrupt x4A. This interrupt is activated by interrupt x70 when the clock 110 Alarm matches the hours, minutes, and seconds values of the clock time value 102. In one embodiment, this interrupt is set to trigger when the seconds value is x00. Other embodiments of the invention set the interrupt to trigger at times close to 00. Other embodiments of the invention, set the interrupt to trigger at other times (e.g., 00:00:00, nn:00:00, nn:nn:02, where n matches any number).

If the clock 110 Alarm interrupt is selected in step 450, but some other program has already set this alarm, then one embodiment of the resident program determines the lowest common alarm frequency between the current settings and the resident settings and resets the alarm with those settings. This ensures that each interrupt service routine will be serviced. One problem with this interrupt is that another program may acquire this interrupt and prevent the resident program from being invoked.

The third type of interrupt is an interrupt that intercepts calls to the BIOS 150 Time-Of-Day services available through interrupt x1A. Note that in one embodiment, this may be done with interrupts while in another embodiment, a software wrapper intercepts the calls and sends at least some of the calls to the resident program. Intercepting the calls ensures that the interrupt will be received only when the clock 110 is directly requested for the clock date value 102 and/or the clock time value 104.

In the embodiment where step 460 selects one or two of these interrupts, step 460 first determines if another program is using the clock 110 Alarm and the User Timer Tick Counter. If the clock 110 Alarm is not being used, this interrupt is selected. If the clock 110 Alarm is being used by another program, but the User Timer Tick Counter is not used by another program, then the User Timer Tick Counter is used by the fix application 174. However, if the User Timer Tick Counter is used by another program, then intercepting the BIOS 150 Time-Of-Day calls is done by the fix application 174.

In one embodiment, the resident program, in step 480, tests the century value to determine whether the century value is x20. If the century value is x20, then the resident program does not perform anymore steps (e.g., the resident program removes the interrupts set in step 460). If the century value is x19, then the resident program proceeds to the step 490.

Scan Test

This section describes the scan test application 172. First the user interface features of one embodiment of the invention are described. (Figure 5 through Figure 7 show elements of the user

interface.) Next, one embodiment of a method of scanning files on a computer system 100 is described.

The following user interface is but one example of a user interface for the scan application 176. Other embodiments of the invention use other interfaces, such as a text only interface.

Figure 5 illustrates one embodiment of a user interface to a program for scanning files for the year 2000 problem. Figure 5 illustrates the scan window 520. The scan window 520 presents the scan test application 176 information to the user. The scan window 520 includes a number of buttons in the menu bar 510. The menu bar 510 allows a user to begin a scan, customize a scan, obtain help information and exit the scan application 176.

Figure 6 illustrates one embodiment of the customize window 610 presented to the user when the user selects customize from the menu bar 510. The customize window 610 includes a number of subwindows: a directory window 620, a files to scan window 630, a file types to skip window 640 and a date formats window 650.

The directory window 620, the files to scan window 630 and the file types to skip window 640 allow the user to better reduce the number of files that need scanning so as to target the files most likely to include year 2000 problems. In particular, the directory window 620 allows the user to select which directory(ies) to be scanned. The file types to skip window 640 lists all of the file types to skip during a scan. In the example of Figure 6, the user has requested that the executable files ("*.EXE"), dynamic link libraries ("*.DLL"), "*.COM" files, "*.OBJ" files and "*.SWP" files not be scanned. In this way, the user has targeted primarily data files for year 2000 scan testing.

The date formats window 650 includes a list of problem date formats. A problem date format includes a definition of the pattern of a problem date. Some characters in a problem date format represent variables (such as the month name). The remaining characters are literals. The characters that represent variables include:

" " - matches one or more white space characters;

"D" - matches the name or abbreviation of a day, e.g. "Monday" or "Mon." or "MON";

"d" - matches a day of the month, i.e., 1...31;

"M" - matches the name or abbreviation of a month, e.g. "January," or "jan." or "JAN";

"m" - matches the number of a month, i.e., 1...12;

"Y" - matches a two digit year in the range 90...99; and

"y" - matches a two digit year in the range 32...99 (this range ensures that "y" will not be mistaken for a day or month).

Other embodiments of the invention include other variables. In one embodiment, the variables are case insensitive and match ASCII and EBCDIC characters.

An example problem date format is "M d, y" which matches a text string including the name or abbreviation of the month, followed by one or more white space characters, followed by a one or two digit day of the week, followed by a comma, followed by one or more white space characters, followed by a two-digit year in the range 32...99. The use of the date formats is described in greater detail below.

The customize window 610 also allows the user to specify a maximum number of file hits during a scan. This is because it may be desirable for a user to just note the files affected and limit the information per file (e.g., 10 dates maximum).

Figure 7 illustrates some example results of a scan test for the year 2000 problem. The results are presented in the scan results window 710 when the user selects the scan button in the menu bar 510.

The scan results window 710 includes a status window 720, a status bar 780, and a scan output window 730. The status window 720 indicates the status of a scan operation. In the example of Figure 7, the scan operation has completed and the status window 720 indicates that eleven files were scanned. The status bar 780 indicates the percentage of the data scanned out of the total data to be scanned.

The scan output window 730 includes the results of a scan for each of the scanned files. In the example of Figure 7, each scanned file result includes a file name 732, a position 740, a line number 750, a problem date format 760, and a problem date 770. The file name 732 corresponds to the name of the file being scanned. The position 740 indicates the character position in the line where the start of the problem date occurs. The line number 750 indicates the line number in the file in which the problem date was found (in this case, the file has a binary format so the line number is irrelevant). The problem date format 760 shows which problem date format, defined in the date formats window 650, a particular problem date matches. The problem date 770 indicates the exact text in the file having the problem date.

Figure 8 illustrates one embodiment of a method of scanning for the year 2000 problem. The purpose of the embodiment of Figure 8 is to scan files in the computer system 100 that have

potential year 2000 date problems. Each file selected in the customize window 610 is scanned using the embodiment shown in Figure 8.

At step 810, the type of the file is detected. In one embodiment, this is done by comparing the extension name of the file with a known list of file type extensions. For example, if the file name is "PATPOLSH.EXE," then the "EXE" extension is matched to an executable extension. If the file type is unknown, then step 830 through step 834 are executed. If the file type is a known data type, then step 840 through step 844 are executed. If the file type is a program or other known executable file, then step 850 and step 854 are executed. If the file type is software source code, then step 860 and step 862 are executed.

Assuming the file type is unknown, then at step 830, a test is made to determine whether all the characters in the file have been examined. If all the characters have been examined, then the scanning terminates for the file. If not all the characters have been examined, then at step 832 the next character is tested, in step 832, to determine if that character could be the start of a problem date, as defined in the problem date formats. If the character could be the start of a problem date, each subsequent character is tested until a problem date is detected or no problem date is detected. If a problem date is detected, then, at step 834, the location, format and problem date value is stored. This information can then be displayed in the scan results window 710. Also the pointer to the next character is positioned after the present character. If the character could not be the start of a problem date, or subsequent characters show that the character is not the start of a problem date, the next character pointer is positioned at the next character after the present character.

To increase the effectiveness of the scanning of files, known data type files can be scanned on a per field basis. This allows the scan application 172 to skip fields in the file. Assuming the file type is a known data file type, then at step 840, a test is made to determine whether all of the fields within the data file have been examined. Examples of a known data type of file are a spreadsheet file, such as an Excel™ spreadsheet file, and a database file, such as a FileMaker Pro™ database file. A field in such a file can be a calculation field, a data field, or a comparison field. If all the fields in the file have been examined, then the scanning of the file ends. If fields remain, then at step 842, a test of the field is made to determine whether the field includes a value corresponding to a problem date format. Thus, step 842 is similar to step 832. If the field does not correspond to a problem date format, then the step 840 is executed for the next field in the file. However, if the field examined at step 842 corresponds to a problem date format, then the location, format and date value of the field is stored. The information stored in step 844 can then be output to the scan results window 710.

To provide additional information about potential problem applications 170, the scan application 176 includes a test for known problem programs. Step 850 and step 854 note the existence of these known problem programs. At step 850, the name and version number of the program are compared against a database of known problem programs. If the file being scanned does not correspond to one of these known problem programs, then the scanning ends for that file. If the file being scanned corresponds to one of the known problem programs, then step 854 is executed. In step 854, the name and version of the known problem program is stored. The information stored in step 854 can then be displayed in the scan results window 710.

To provide additional information, the scan application includes a test for program source code. Step 860 and step 862 note the existence of program source code files that may contain the problem. At step 860 program variables are identified using parsing algorithms for known source code languages (for instance, Visual Basic, C, Pascal, Java). If any variables have a date-like name (e.g., contain the strings "date" or "year") or are used in a potentially date sensitive context (e.g., a variable is assigned the results of a date request from the computer system, or a variable is assigned a value that is determined from a date-like name variable), then step 862 is executed. In step 862 the file name and variable names are stored. The information stored in 862 can then be displayed in the scan results window 710.

Thus, the above has described one embodiment of a method of detecting files in a computer system 100 with potential year 2000 problems.

Additional Embodiments

The following describes some additional embodiments of the invention. As noted previously, multiple embodiments of the invention exist, however, all of these embodiments address year 2000 problems in personal computer systems. The following embodiments show how the invention can be used in other ways to address year 2000 problems.

In one embodiment, the computer system 100 is coupled to a network, such as a Network™ network, available from Novel, Inc. In other embodiments, the computer system 100 is coupled to other computer systems via the internet, a direct modem connection, or a wireless connection.

The computer system 100 can then execute the test application 172 on itself as well as other computer systems on the network. In one embodiment, as an additional step in the testing

process, the test application 172 checks the type of the BIOS 150 and compares the type against a known list of BIOS types. In one embodiment, if the BIOS type is found in the list, then the appropriate action for that type of computer system is retrieved. For example, if the type of BIOS 150 indicates that the fix application 174 should be installed, then the computer system 100 can install the fix application 174 on the networked computer, without having to execute the other test steps. In one embodiment, for unknown BIOS types, after running the tests, the test application 172 updates its list of known BIOS types. In other embodiments, other information is kept in the list, such as the operating system 160 being run, and the type of computer (if this information is available). By keeping the list of information, a manager of information services can quickly assess year 2000 problems in multiple computers on the network. From the list the fixes can be installed in the networked computers without necessarily having to test each machine.

In one embodiment of the invention, the computer system 100 also performs a batch scan test of files stored in the networked computers. In another embodiment, the computer system 100 performs a scan test on the back up files from the networked computers. Scanning the back up files allows the scan application 176 to run its scans without having to directly access the files on the networked computers.

In another embodiment of the invention, the embodiment of Figure 2 is extended to also test whether the leap day, February 29, 2000, is properly handled in the computer system 100. In another extension to the embodiment of Figure 2, a test is made to ensure that the computer system 100 transitions from December 31, 2000 to January 1, 2001.

The above description provides example embodiments of the invention. However, the scope of the invention is determined by the following claims.

Summary of Other Aspects of Invention

Aspect 1. A method of automatically detecting a date problem in a computer system, said computer system including a processor, a memory, a clock, and a non-volatile memory, said processor being coupled to said memory, said clock and said non-volatile memory, said non-volatile memory storing a clock date value and a clock time value, said clock date value including a clock century value and a clock year value, said method comprising:

storing a first date value and a first time value corresponding to said clock date value and said clock time value, respectively, at a first time;

changing said clock date value to December 31, 1999;

changing said clock time value to a value less than or equal to 11:59:59 PM:

testing said clock century value and,

if said clock century value indicates twenty, then indicating that said date problem does not exist, and

if said clock century value does not indicate twenty and said clock year value does not indicate 99, then indicating that said date problem does exist;

changing said clock date value to correspond to said first date value; and

changing said clock time value to correspond to said first time value.

Aspect 2. The method of aspect 1 wherein said computer system uses a first bus architecture and wherein said method further comprises testing said computer system to determine a bus type of said first bus architecture and wherein said clock century value is accessed at a location in said non-volatile memory determined from said bus type.

Aspect 3. The method of aspect 2 wherein said first bus architecture includes one of a Micro Channel Bus Architecture bus and a Industry Standard Architecture bus.

Aspect 4. A method of aspect 1, 2 or 3 wherein said computer system uses a first bus architecture and wherein said method further comprises testing said computer system to

determine a bus type of said first bus architecture and wherein said clock century value is accessed at a location in said non-volatile memory determined from said bus type.

Aspect 5. The method of any preceding aspect further comprising a step of testing said computer system for an existence of said clock and only executing other steps in said method if said clock exists.

Aspect 6. The method of any preceding aspect wherein said clock time value includes a clock seconds value and prior to said storing said first date value and said first time value, testing said clock seconds value, if said clock seconds value is greater than a predetermined value, then waiting until said clock seconds value changes to zero before executing said storing said first date value and said first time value.

Aspect 7. The method of any preceding aspect wherein said changing said clock time value to correspond to said first time value includes storing a second time value in said clock time value, where said second time value corresponds to said first time value plus a period of time used to execute said method.

Aspect 8. The method of any preceding aspect wherein said clock and said non-volatile memory are included in an integrated circuit having similar functions to an MC146818A integrated circuit, and wherein said integrated circuit includes a two nibble binary coded decimal (BCD) years register, a two nibble BCD months register, a two nibble BCD days register, a two nibble BCD hours register, a two nibble BCD minutes register a two nibble seconds register, wherein said clock is operating on a twelve hour format, wherein said changing said clock date value to December 31, 1999 includes storing x99 in said two nibble BCD years register, storing x12 in said two nibble BCD months register, storing x31 in said two nibble BCD days register, and wherein said changing said clock time value to a value less than or equal to 11:59:59 PM includes storing x11 in said two nibble BCD hours register, storing x59 is said two nibble BCD minutes register, and storing a value less than or equal to x59 in said two nibble BCD seconds register.

Aspect 9. A method of detecting a date problem in a computer system, said computer system including a processor, a memory, a clock, a non-volatile memory, said processor being coupled to said memory, said clock, and said non-volatile memory, said non-volatile memory including a clock date value, said clock date value including a clock century value, said method comprising:

said computer system causing said clock century value to indicate twenty;

said computer system restarting said computer system; and

testing said clock century value after restarting said computer system, if said clock century value does not indicate twenty, then said computer system indicating that said date problem exists.

Aspect 10. The method of any preceding aspect wherein said non-volatile memory includes a clock time value, said method further comprising, if said clock century value is x20, then,

causing said clock date value to be less than January 1, 1980;

restarting said computer system a second time; and

testing said clock century value after restarting said computer system said second time and if said clock century value is x20, then indicating said computer system will correct said date problem upon restarting said computer system, otherwise indicating said date problem exists.

Aspect 11. A method of aspect 10 wherein said causing said clock date value to be less than January 1, 1980 includes setting said clock date value to be January 1, 1900.

Aspect 12. The method of aspect 9, 10 or 11 wherein said computer system includes a Basic Input Output System (BIOS), said clock date value includes a clock year value, said method further comprising:

causing said clock year value to be a first year value, said first year value indicating less than the eightieth year; and

testing said clock year value after restarting said computer system, if said clock year value does not correspond to said first year value, then indicating said BIOS should be replaced.

Aspect 13. The method of aspect 12 wherein said clock year value does not correspond to said first year value if said clock year value does not equal said first year value.

Aspect 14. The method of aspect 9, 10, 11, 12 or 13 wherein said non-volatile memory also includes a clock time value, said method further comprising the following steps to be performed prior to causing said clock century value to indicate twenty:

setting said clock time value to be less than or equal to 11:59:59 PM;

setting said clock date value to be December 31, 1999;

waiting for a period of time; and

testing said clock century value prior to said restarting said computer system.

Aspect 15. The method of aspect 14 where said computer system includes a Basic Input Output System (BIOS) and where if in testing said clock century value prior to restarting said computer system, said clock century value is nineteen, then;

restarting said computer system a second time and testing said clock century value after restarting said computer system said second time, if said clock century value is twenty after restarting said computer system said second time, then indicating that said BIOS corrects said date problem after restarting said system.

Aspect 16. The method of aspect 14 where if in testing said clock century value prior to restarting said computer system, said clock century value is twenty, and if in testing said clock century value after restarting said computer system, said clock century value is twenty, then indicating that said date problem does not exist.

Aspect 17. The method of aspect 14 wherein said clock date value includes a clock year value, and wherein said computer system includes a Basic Input Output System (BIOS), said

method further comprising testing said clock year value and indicating said BIOS should be replaced if said clock year value is not zero.

Aspect 18. A method of fixing a date handling problem in a computer system, said computer system including a processor, a non-volatile memory and a clock, said processor being coupled to said non-volatile memory and said clock, said non-volatile memory having a century field and a year field, said clock for modifying said century field and said year field according to the time, said method comprising the steps of:

selecting an interrupt from a set of predetermined interrupts;

setting said interrupt to cause said clock to interrupt said processor at a predetermined time;

receiving said interrupt; and

after receiving said interrupt, testing said century field and said year field, if said century field does not indicate twenty and said year field indicates a year less than a predetermined value, then changing said century field to equal twenty.

Aspect 19. The method of aspect 18 where said predetermined value is x80.

Aspect 20. The method of aspect 18 where said century field represents numbers in binary coded decimal and where said testing said century field includes comparing a value in said century field with x20.

Aspect 21. The method of aspect 18, 19 or 20 where said non-volatile memory includes a seconds field and where said predetermined time is when said seconds field indicates zero.

Aspect 22. The method of aspect 18 where said non-volatile memory includes an hours field, a minutes field and a seconds field and where said predetermined time is when said hours field indicates twelve, said minutes field indicates zero, and said seconds field indicates zero.

Aspect 23. The method of aspect 22 wherein said predetermined time is when said hour field indicates twelve AM.

Aspect 24. The method of any of aspects 18 to 23 wherein said computer system includes a Basic Input Output System (BIOS), said BIOS supports a BIOS alarm that interrupts said processor at a specific time, and where said setting said interrupt includes setting said BIOS alarm to interrupt said processor at said predetermined time.

Aspect 25. The method of any of aspects 18 to 23 wherein said computer system includes a Basic Input Output System (BIOS), where said BIOS supports a user settable tick count alarm, said user settable tick count alarm interrupting said processor approximately eighteen times per second, and where said setting said interrupt includes setting said user settable tick count alarm, and where said testing said century field and said year field is performed after receiving a predetermined number of interrupts.

Aspect 26. The method of aspect 25 wherein said predetermined number of interrupts is sixteen.

Aspect 27. The method of aspect 25 or 26 wherein said computer system includes a Terminate and Stay Resident (TSR) program and wherein said steps of said method are performed by said TSR program.

CLAIMS

1. A computer system comprising:

means for accessing each of a plurality of stored files;

means for scanning an accessed file to ascertain whether the file uses, or is of a type which uses, one or more predetermined problem date formats having no century indicator; and

means for compiling a list of files having the one or more predetermined problem date formats.

2. A computer system according to claim 1, wherein the scanning means is operable to identify files corresponding to predetermined applications, or to data files for therefor, known to use a problem date format.

3. A computer system according to claim 1 or 2, wherein the scanning means is operable to scan the file for one or more information strings or fields matching one or more predetermined problem date formats having no century indicator.

4. A computer system according to claim 3, wherein the scanning means is operable to scan the file for a data field type matching a predetermined date field type representing a date field in a data file which does not include a century representation.

5. A computer system according to claim 3 or 4, further comprising means for defining the one or more predetermined problem date formats for the scanning means, by identifying the location in an information string or field of day information, month information, and year information.

6. A computer system according to claim 5, wherein one of the definitions includes "M d, y", said definition matching a text string including the full or truncated name of a month, followed by one or more space characters, followed by a one or two

digit day value, followed by a comma, followed by one or more space characters, followed by a two digit year value.

7. A computer system according to claim 5 or 6, wherein at least one of the definitions includes date information in the order day, month and year.

8. A computer system according to any of claims 1 to 7, wherein the scanning means is operable to scan the file for variables with date-like names if the file is a source code file.

9. A computer system according to any of claims 1 to 8, wherein the scanning means is operable to scan the file for variables with date sensitive context if the file is a source code file.

10. A method of automatically processing a plurality of computer files stored on a memory device, comprising:

scanning each file to ascertain whether the file uses, or is of a type which uses, one or more predetermined problem date formats having no century indicator; and

compiling a list of files having the one or more predetermined problem date formats.

11. A method of identifying a subset of files from a set of files, said subset of files including files having potential date problems, said method comprising the steps of:

scanning each file of said of files, including:

(a) searching said each file for one or more strings matching a predetermined problem date format, said predetermined problem date format corresponding to a date representation without a century indicator; and

(b) if said each file includes one or more strings matching said predetermined problem date format, then including said each file in said subset of files.

12. A method according to claim 11, wherein said scanning each file further comprise:

searching said each file for one or more fields corresponding to one or more data field types, each of said one or more data field types matching a predetermined date field type, each predetermined data field type indicating a date field representation in a data file that does not include a century representation; and

if said each file includes one or more fields corresponding to one or more data field types, then including said each file in said subset of files.

13. A method according to claim 11 or 12, wherein said scanning each file further comprises including said each file in said subset of files if said each file is an application of a predetermined application type, said predetermined application type indicating an application having a year 2000 problem.

14. A method according to claim 11, 12 or 13, further including receiving a definition of said predetermined problem date format, where said definition of said problem date format indicates a location of a day value, a month value, and a year value in a string of characters.

15. A method according to claim 14, wherein said definition includes "M d, y", said definition matching a text string including the name of a month, followed by one or more white space characters, followed by a one or two digit day of the week, followed by a comma, followed by one or more white space characters, followed by a two-digit year.

16. A method according to claim 11, 12, 13, 14 or 15, wherein said searching each said file includes searching for variables with date-like names if said each file is a source code file and including said each file in said subset of files if said each file includes variables with date-like names.

17. A method according to any of claims 11 to 16, wherein said searching said each file includes searching for variables with date sensitive context if said each file is a source code file and including said each file in said subset of files if said each file includes variables with date sensitive context.

18. A method or system for identifying computer files having a year 2000 problem, substantially as hereinbefore described with reference to any of Figs. 1, 5, 6, 7 and 8 of the accompanying drawings.

19. A computer system comprising:

real-time clock-means for maintaining date information, the date information including century information;

means for automatically testing whether the real-time clock means is operative to update the century information correctly at a century change; and

means responsive to the result determined by the first means, for selectively implementing a fix to correct the century information at the century change.

20. A computer system according to claim 19, wherein the computer system comprises processing means and memory means accessible directly or indirectly by said processing means, wherein said first and second means comprise program data stored at least partly in the memory means.

21. A computer system according to claim 19 or 20, wherein the implementing means is operable to install a fix program in the computer depending on the result of the test.

22. A computer system according to claim 19, 20 or 21, comprising means for storing a plurality of files, and means for scanning the files to identify files which include date information having a potential century date problem.

23. A computer system according to claim 22, wherein the scanning means is operable to identify files having a date format with no century indicator.

24. A computer system comprising:

a real-time clock circuit including a date storage location, said date storage location including a century storage location, said century storage location including a century value;

a test application coupled to receive said century value to determine whether said computer system updates said century value to indicate the twentieth century at midnight on December 31, 1999;

a fix application coupled to change said century value, said fix application being responsive to a result generated by said test application, said result indicating whether said computer system updates said century value to indicate the twentieth century at midnight on December 31, 1999;

a processor executing said test application and said fix application; and

a memory storing portions of said test application and said fix application.

25. A computer system according to claim 24, further comprising:

a plurality of files; and

a scan application coupled to scan said plurality of files and to output a plurality of results, some results of said plurality of results including a file identifier and one or more date problem indicators, said file identifier corresponding to one of said plurality of files, and each problem date indicator indicating a location of a string of date corresponding to a date which does not include a century indicator.

26. A computer system according to claim 25, wherein some of said plurality of files correspond to a set of known data file types, said set of known data file types corresponding to a set of known problem field types, each known problem field type corresponding to a field representing a date without indicating a century, some of said

plurality of results including a reference to a known problem field type found in a corresponding file.

27. A computer system according to claim 25 or 26, wherein said plurality of files include a plurality of applications, some of said plurality of results indicating a known problem application corresponding to one application of said plurality of applications, said known problem application using date values without correctly using a century value.

28. An automatic date correction method for a computer having real-time clock means for maintaining date information including century information, the method comprising:

executing a first test program for testing automatically whether the real-time clock means is operative to update the century information at a change of century; and

selectively executing a second program in response to the result of the test in the first program, for implementing a fix to correct the updating of the century information at a century change.

29. A method according to claim 28, wherein the second program is operative to install a fix program in the computer.

The Patent Office

45

Application No:	GB 9715829.9	**Examiner:**	Matthew Gillard
Claims searched:	1-18	**Date of search:**	26 August 1997

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A AEX

Int Cl (Ed.6): G06F 11/00, 11/22, 11/26

Other: On-line: WPI, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	EDGE: Work-Group Computing Report, v8, p22(1), 24 Feb 1997.	-
X	Datamation, v42, n1, p49(3), 1 Jan 1996, D Baum. See "IMPACT ANALYSIS" and "REMOVING THE PROBLEM" (paras 1-3 thereof).	1-4, 8-13, 16 & 17

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.